2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

A Hierarchical Decoupled Approach for Multi Robot Motion Planning on Trees

Ellips Masehian and Azadeh H. Nejad

Abstract – **In this paper, the multi robot motion planning problem is solved through a decoupled approach, where a new algorithm for prioritizing the robots moves is developed. Assuming that the workspace is mapped into a tree graph and the initial and final configurations of robots are known, the robots' shortest start-to-goal paths on the tree are calculated independently. Then, a new rule-based prioritization scheme is applied in two phases: (a) Path Prioritization, which determines which robot can directly move along its shortest path and which robot should deviate from it, and (b) Motion Prioritization, which decides the order of robots' sequential moves. Furthermore, an algorithm is presented for minimizing the number of moves by adding extra vertices to the tree.**

I. Introduction

The Multi Robot Motion Planning (MRMP) problem is gaining increased attention as many real-world complex applications require collective intelligent behaviors exhibited by multiple robots. The basic MRMP problem is to find start-to-goal paths for a number of robots moving in a space with obstacles, such that they do not collide with obstacles or each other. Space is the most limiting constraint in a typical MRMP problem: often, because of lack of sufficient space around robots, they cannot reach their goals without obstructing each other's way, causing *deadlocks*. Deadlocks are situations where two or more robots intercept each other's motions and are prevented from reaching their goals. This happens generally in narrow passageways where robots cannot pass by each other.

A. Related Work

The MRMP problem is proved to be NP-hard [1], and is solved by two general approaches: *Centralized* planning and *Decoupled* planning. In the Centralized planning, all m robots in the scene are regarded as a single multi-part robot with m independent bodies that are not necessarily connected to each other. This (virtual) compound robot has a degree of freedom equal to the sum of the degrees of freedom of all part-robots. Thus, collisions of r_i robots (i = 1, …, m) is interpreted as self-collisions of the compound robot. This approach, however, suffers from high time and memory requirements for large degrees of freedom [2].

The Decoupled approach first plans the motion of each robot individually while ignoring the existence of the other robots, and then tries to combine the resulting paths by resolving possible collisions between the paths. Decoupled planning can handle problems with large number of robots and is implemented more extensively in the literature. Collision resolution is performed through two techniques: *Velocity Tuning*, and *Prioritizing*. In the Velocity Tuning technique, each robot either stops or decelerates its motion to avoid colliding with other robots [3]. The Prioritization technique assigns priorities to each robot and computes the paths of the robots based on the order of these priorities [4].

Many different approaches are proposed for assigning priorities to robots: Buckley's work is among the first ones, in which through applying a heuristic algorithm, higher priorities are assigned to robots that can move to their goals directly [5]. In [6] an approach is described that uses fixed order of priority and chooses random alternative routes for robots with lower priorities. A randomized hill-climbing search is used in [7] to assign the priorities of robots while applying A* search to compute the optimal path of each robot.

In [8] a fully distributed cooperative path planning is done on local sections of the time-space configuration space, where dynamic conflicts between the robots are solved by the heuristic adjustment of priority values. A decentralized motion control system of a team of mobile robots is developed in [9], which leads each robot to its individual goal by online modification of pre-computed navigation functions in order to satisfy formation constraints. The method handles a limited number of robots, and is tested only on sparsely located circular obstacles.

Although methods dealing with continuous spaces have some advantages, but they suffer from increased computational burden for high number of robots. To overcome this, the continuous Configuration space has been discretized into a graph or tree in a number of works. This is done meant for downsizing the C-space and implicitly modeling workspace obstacles (by locating graph nodes and arcs inside the free C-space), and thus reducing computations. Another advantage of discretizing the space is the possibility of applying techniques such as sequencing for motion planning.

In [10] the possibility of reaching destinations of connected subgraphs is studied, by simplifying the MRMP between some predefined subgraphs named stacks, halls, rings, and cliques. In [11] the coordination problem of multiple robots on a network is solved by introducing a social law on it which all the robots should obey. The social law is derived from a routing of the graph underlying the network

A multi-phase approach to the planning problem by using a topological graph and spanning tree representation of a tunnel or corridor environment is presented in [12], which plans trajectories for a number of robots proportionate to the number of leaves of the spanning tree. However, a drawback of the method is that the construction of the spanning tree is time consuming since it is not unique, and with selecting different trees, different number of robots would be dealt with. Also, the aggregate path length is considerably long compared to a standard decoupled method.

Authors are with the Faculty of Engineering, Tarbiat Modares University, Tehran, Iran. Corresponding author's e-mail: masehian@modares.ac.ir.

B. The Current Work

When the workspace is mapped into a graph (such as the Voronoi diagram), the robots are required to move only along predefined routes (i.e. graph edges), implicitly avoiding the obstacles existing in the workspace. The MRMP on graphs has real-world applications in maze-like environments, indoor rooms or offices connected with corridors, AGV routes in plants, multi-story parking lots, railway networks, etc.

Graphs can be categorized into two general classes: *cyclic*, and *acyclic*: cyclic graphs have loops and provide alternative ways to access a specific node, and therefore are more convenient for planning the motions of multiple robots. On the other hand, acyclic graphs (i.e. trees) do not have any loops, and the shortest path between every pair of nodes is unique. As such, trees provide fewer options and less maneuverability for the robots, and are much harder to solve than cyclic graphs. Actually, MRMP on trees can serve as a basis for MRMP on cyclic graphs, and the solution of an MRMP on a tree is also valid for the MRMP problem on a cyclic graph which is the superset of that tree. In view of this, and considering its NP-hardness, we found the MRMP problem on trees a worthwhile and challenging problem to be solved efficiently.

A related issue to the MRMP is that what topology a tree should have in order to be solvable. A *Solvable Tree* is a tree that allows the transition of any initial configuration of robots to a final configuration via their moves on arcs. The topologies of solvable trees are proposed in [13]. The solvability analysis of a problem can determine if the given MRMP problem has a solution, without explicitly solving it.

By designing the current decoupled method we intended to overcome the scalability problem of non-graph-based MRMP methods such as [8] and [9] (which hardly deal with more that 10 robots), as well as improving the collective path length of graph-based methods like [11] and [12] (which plan many redundant moves and take long time). In fact, as far as we tested the proposed method, it can solve MRMP problems with hundreds of robots in quite reasonable times.

The next Section of the paper provides some definitions and assumptions of the algorithm and briefly describes its steps. Section III and IV explain in detail the procedures of path prioritization and motion prioritization, respectively. In Section V the tree is enhanced to minimize the number of moves required for deadlocks resolution. Discussions and conclusions are presented in Section VI.

II. Algorithm Assumptions and Overview

Before proceeding, some terms and symbols are introduced: Let $T = (V, E)$ be a tree, with the set of vertices V and set of edges E. the number of vertices connected to the vertex v is called its Degree, $d(v)$. The *Leaves* of a tree, $L(T)$, are vertices with $d(v) = 1$, and the *Internal Vertices* of a tree, $I(T)$, are vertices with $d(v) > 1$. The set of vertices connecting v and u (inclusive) is shown by $Path(v, u)$.

The *Origin* $\mathrm{O} \in \{v \mid d(v) \geq d(u), \forall u \in V\}$ is defined as the vertex with maximum degree in the Tree. If not unique, it is selected such that the maximum Level of all vertices is kept minimal. The *Level* of a vertex $l(v)$ is the minimal distance of vertex v from O (i.e. $Dist(v, \mathrm{O})$). The tree can be searched in a linear time complexity to decide its Origin. The origin is taken as the *Root* of the tree, and the Levels of all vertices are determined in relation to the Origin.

s_i and g_i are the start and goal of robot r_i, and SP_i is the sequence of vertices on the shortest path of r_i from s_i to g_i in the Tree. Finally, a *Plan* is the sequences of robots' motion on a tree, from initial to final configuration.

The presented algorithm is based on some assumptions:

1. The free space is represented by a tree, which is finite, connected, planar, and undirected.
2. Initial and final locations of all robots lie on the tree and are known.
3. All robots share the same tree and can only reside on vertices of the tree, and can only move along edges of the tree. Each vertex can accommodate only one robot at a time, and only one robot can pass along an edge at a time.
4. A robot at vertex v can move to its neighboring vertex u only if u is unoccupied; i.e. moves are sequentially. Robots occupying other vertices in the graph do not affect this move. Nevertheless, after finalizing the Plan, this assumption can be relaxed by advancing as much as possible the motions of robots which do not form deadlocks with already moved robots, thus introducing concurrency of motions.

A. Overview

As stated earlier, deadlocks in the decoupled planning approach can be resolved by introducing a *priority scheme*, which determines the order in which the motions of the robots are planned. In our method, the priority scheme is implemented hierarchically through two steps: prioritization of robots *paths*, and prioritization of robots *motions*. This is done since these two types are different in nature: a robot with priority of path planning has the right to move *straightforwardly* toward its goal (i.e. travel along its shortest path), whereas a robot with priority of motion planning has the right to move *earlier* (either to its goal or a temporary node).

For instance, the robot 1 in Fig. 1(a) (located at S1) and robot 2 in Fig. 1(b) (located at S2) have higher path planning priorities. Yet, in both figures, the motion priority is given to the robot 1. However, in a problem like Fig. 1(c), the path planning priorities of both robots are equal, but the motion priority of robot 2 is higher than that of robot 1.

The outline of the algorithm is as follows:

Step 1: For each robot the shortest start-to-goal paths are calculated independently. Based on robots' starts and goals, they are categorized into two classes: robots that can move

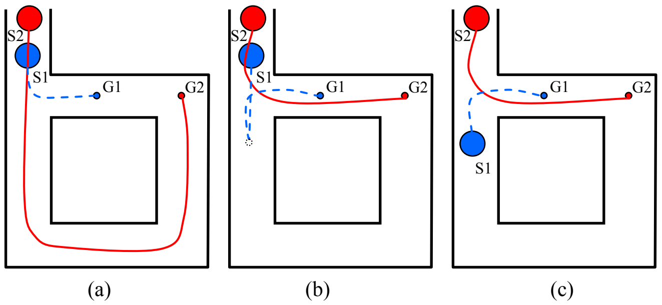


Fig. 1. Path and Motion priorities in a sample problem (from [7]).

along their Shortest Paths, and robots that must (temporarily) deviate from their Shortest Paths in order to let the robots of the first class reach their goals first.

Step 2: The robot's motions are then prioritized through two phases: Rough, and Exact Motion Prioritization.

Step 3: Aiming to minimize the robots' total traversed distance and resolve possible deadlocks, the temporary stations of deviating robots are determined.

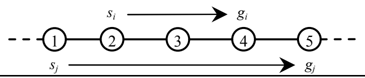
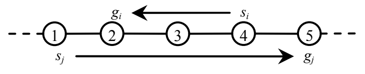
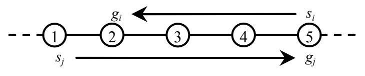
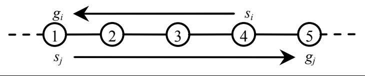
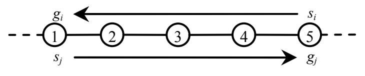
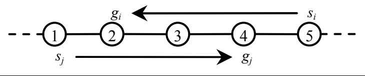
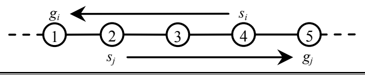
Step 4: The final Plan is generated based on the previous steps.

III. Path Prioritization

In the presented algorithm, shortest start-to-goal paths are calculated independently for each robot. If some paths intersect at certain vertices, then some robots may block other robots' motions to their goals, which will lead to probable collisions. Different cases of blockings (i.e. deadlocks) can occur for two robots moving at 'same' or 'opposite' directions in a tree, as illustrated in Table I.

The relative direction of each pair of robots is determined based on the order of vertices visited by each robot. For instance, in Case 1, $SP_1 = \{1\to2\to3\to4\to5\}$ and $SP_2 = \{2\to3\to4\}$, and so $SP_2 \subset SP_1$. Or in Case 3, $SP_1 = \{1\to2\to3\to4\to5\}$ and $SP_2 = \{5\to4\to3\to2\}$, and so $SP_2 \subset \neg SP_1 \wedge s_2 = g_1$. The symbol '¬' indicates the path in reverse order.

TABLE I
Deadlock Situations for two robots

Case	Direction	Type of deadlock situation
1	same	
2	opposite	
3	opposite	
4	opposite	
5	opposite	
6	opposite	
7	opposite	

Prioritization of paths determines which robot has the privilege to move to its goal without deviating from its Shortest Path, and which robot should sacrifice the local minimality of its motion for the sake of minimizing the total moves in the Plan. In other words, in path prioritization, a class of robots called Non-Deviating Robots (NDRs) are decided to follow their Shortest Path, while another class called Deviating Robots (DRs) have to plan their paths by taking into account the already planned paths of NDRs. DRs have higher priorities for moving than NDRs: that is, only DRs are deviated from their Shortest Paths and make enough room for NDRs to move straightforwardly to their goals.

The decision to classify a robot as Deviating or Non-Deviating is based on its initial (start) and final (goal) configurations. A robot can occupy either a Leaf or an Internal Vertex as its start and goal positions. Four different situations may happen regarding the locations of starts and goals of a pair of robots: (1) a robot's start might be the goal of any other robot (case S=G), (2) a robot's goal might be the start of any other robot (case G=S), (3) a robots' both start and goal are the goal and the start of one (or two) other robot(s) (case S≡G), or (4) none of the above (case S≠G). Based on the type of the start and goal vertices of robots (i.e. Leaf or Internal Vertex), 16 different states may occur, as indicated in Table II.

TABLE II
Different States for a Robot's Initial and Final Configurations

Goal on / Start on	Internal vertex				Leaf			
	S=G	G=S	S≡G	S≠G	S=G	G=S	S≡G	S≠G
Internal vertex	1	2	3	4	9	10	11	12
Leaf	5	6	7	8	13	14	15	16

For example, State 6 represents the instance where the start of a robot (say r_i) is on a Leaf, and its goal is on an Internal Vertex, which is itself the start of another robot (say r_j).

Since Internal Vertices are the only places in the tree where robots can shift from a leaf to another, they should be vacant as early and as much as possible. Also, robots located on the goals of other robots should deviate from their shortest paths, regardless of their destination. Conversely, since robots with starts on Leaves do not block other robots' paths (unless the Leaf is itself a goal), they need not make extra moves other than their shortest paths. By implementing the above intuitive principles, the robots can be classified into two general classes, DRs and NDRs, as shown in Table III.

TABLE III
Classes of Robots Categorized Based on States

Robots in States	1, 2, 4, 5, and 10	are Deviating (DR)
Robots in States	6, 8, 9, 12, 13, 14, and 16	are Non-Deviating (NDR)
Robots in States	3, 7, 11, and 15	cannot be determined

The States 3, 7, 11 and 15 in the Table II refer to the case S≡G, where the path planning priorities of involved robots are equal, and so the robots' classes (i.e. DR or NDR) cannot be determined. Instead, their class should be decided with respect to the already determined classes of other robots, and regarding their created deadlock situations, as indicated in Table IV (compare to Table I).

TABLE IV
Classes of Robots Classified Based on Deadlock Situations

Case	Deadlock situation	The Deviating Robot
1	$SP_i \subset SP_j$	r_i
2	$SP_i \subset \neg SP_j \wedge s_i \neq g_j \wedge s_j \neq g_i$	
3	$SP_i \subset \neg SP_j \wedge s_i = g_j$	
4	$SP_i \subset \neg SP_j \wedge s_j = g_i$	r_i if r_j is NDR or r_j if r_i is NDR
5	$SP_i = SP_j \wedge s_i = g_j \wedge s_j = g_i$	
6	$g_i \in SP_j \wedge g_j \in SP_i$	
7	$SP_i \not\subset SP_j \wedge SP_j \not\subset SP_i \wedge s_i \in SP_j \wedge s_j \in SP_i$	

The whole process of classifying all robots into DR and NDR classes is explained in Fig. 2. Note that since a robot can form different deadlock situations with different robots, its class may be simultaneously set as both DR and NDR. This contradiction is settled by marking the robot as DR (line 16 in Fig. 2).

The process of classifying robots is illustrated through a sample problem, with 10 robots located on a 13-vertex tree (with A as Origin), as depicted in Fig. 3. Dotted vertices will be discussed later. Considering the shortest start-to-goal paths of robots as shown in Table V, the classes of some robots can be decided according to lines 3 and 4 of the pseudocode.

```
INPUT (a solvable tree for given initial and final configurations of robots)
1  Calculate the Shortest Paths of all robots on the tree
2  Do until the classes (DR or NDR) of all robots are decided
3      Determine the state of each robot (based on Table III)
4      Determine the class of each robot (based on Table IV)
5      For robots with undecided classes do
6          Identify their deadlock situations with robots classified in line 4
7          Determine the class of undecided robots (based on Table V)
8          For robots with still undecided classes do
9              Determine their class with respect to robots classified in line 7
10         end
11         If the class of a robot is still undecided then
12             Categorize its class as a Non-Deviating Robot (NDR)
13         end
14     end
15     If the class of a robot is marked as both DR and NDR then
16         Categorize its class as a Deviating Robot (DR)
17     end
18 end
```

Fig. 2. Pseudocode for classifying robots into DRs and NDRs.

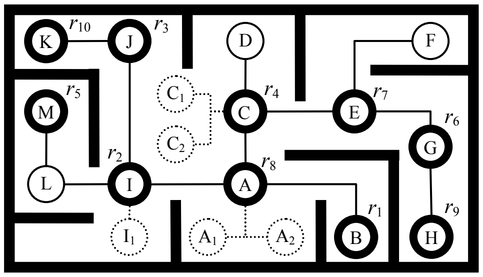


Fig. 3. The initial configuration of 10 robots in a sample workspace.

Further classification of undecided robots is possible by comparing them with the already classified robots (lines 5 to 7), as shown in Table VI. Robot r_5 still remains undecided, and so is compared again for possible deadlocks with the newly-classified robots r_1, r_7, r_8, and r_9 (lines 8 and 9). As a result, it is understood that r_5 forms deadlocks with none of them, and so its class is determined as NDR (lines 11 and 12).

TABLE V
SHORTEST PATHS, STATES AND CLASSES OF ROBOTS IN FIG. 3

Robot	Shortest Path	State	Class
r_1	{B→A→C→E→G→H}	15	undecided
r_2	{I→A→C→E→F}	12	NDR
r_3	{J→I→A→C→D}	9	NDR
r_4	{C→A→B}	10	DR
r_5	{M→L→I→J}	7	undecided
r_6	{G→E→C→A→I→L}	1	DR
r_7	{E→C→A→I→L→M}	11	undecided
r_8	{A→C→E}	3	undecided
r_9	{H→G→E→C→A}	7	undecided
r_{10}	{K→J→I→A→C→E→G}	6	NDR

TABLE VI
DETERMINING THE CLASS OF ROBOTS REMAINED UNDECIDED IN TABLE V

Robot	Compared with	Deadlock situation	Case	Decision
r_1	r_4	$SP_4 \subset \neg SP_1 \wedge s_1=g_4$	4	r_1 is NDR
r_5	$r_2, r_3, r_4, r_6, r_{10}$	none	–	undecided
r_7	r_2	$s_7 \in SP_2 \wedge s_2 \in SP_7$	7	r_7 is DR
r_8	r_2	$SP_8 \subset SP_2$	1	r_8 is DR
	r_4	$s_8 \in SP_4 \wedge s_4 \in SP_8$	7	r_8 is NDR
	r_6	$SP_8 \subset \neg SP_6 \wedge s_8=g_6 \wedge s_6=g_8$	2	r_8 is NDR
	r_{10}	$SP_8 \subset SP_{10}$	1	r_8 is DR
r_9	r_{10}	$g_{10} \in SP_9 \wedge g_9 \in SP_{10}$	6	r_9 is DR

Also, robot r_8 is marked as both DR and NDR due to comparisons with different robots, and therefore is classified as a DR, according to the lines 15 and 16.

IV. MOTION PRIORITIZATION

The deadlock resolution process is generally (but not necessarily) performed in three stages: (1) a DR moves away (deviates) from its Shortest Path and occupies a newly-inserted vertex, (2) an NDR moves from its start to goal directly, and (3) the DR returns to its Shortest Path and reaches its goal vertex. The first stage above is called *Evacuation* stage, and the third is called *Occupation* stage. The second stage simultaneously incorporates both Evacuation and Occupation stages.

For any two robots r_i and r_j, the right given to r_i to evacuate its start earlier than r_j is called its *Evacuation Priority* and is denoted by $EP_i > EP_j$. Alternatively, the right given to r_i to occupy its goal earlier than r_j is called its *Occupation Priority* and is denoted by $OP_i > OP_j$.

Regarding the mutual impacts of multiple DRs and NDRs, setting their motion priorities is a very complex task. In this section, a new Motion Prioritization procedure is developed consisted of two levels: *Rough*, and *Exact*. In Rough Motion Prioritization, a general and tentative motion priority scheme is produced by identifying the number of deadlock situations a robot is involved at its initial and final configurations. On the other hand, Exact Motion Prioritization determines the order of robots in evacuating their starts and occupying their goals with regard to their locations in the Tree.

A. Rough Motion Prioritization

For Rough Motion Prioritization of robots, a binary $m \times m$ matrix called *Deadlocks Matrix* is constructed to summarize all deadlock situations that robots create due to their locations relative to the Shortest Paths of other robots. The rows of this matrix refer to the Shortest Paths of robots, whereas the columns point to robots. Each cell in the *i*-th row and *j*-th column of the Deadlocks Matrix is an ordered pair of 0's or 1's: A '1' as the first element of the pair indicates that the start of the robot *j* is located on the Shortest Path of robot *i*. Alternately, a '1' as the second element indicates that the goal of the robot *j* is located on the Shortest Path of robot *i*. Non-deadlock instances are shown by 0's. The Deadlocks Matrix for the sample problem is formed as Fig. 4.

By summing the first and second elements of the cells in column *j*, the number of deadlocks respectively created by the start and goal of robot *j* with other robots is calculated. A large value in the first element of the sum implies that the start position of the robot *j* blocks the motions of many robots and so must be evacuated earlier. On the other hand, a large value in the second element of the sum implies that the goal position of the robot *j* is on the path of many robots and so must be occupied later.

$$
\begin{array}{c|cccccccccc|}
 & s_1, g_1 & s_2, g_2 & s_3, g_3 & s_4, g_4 & s_5, g_5 & s_6, g_6 & s_7, g_7 & s_8, g_8 & s_9, g_9 & s_{10}, g_{10} \\
SP_1 & 0,0 & 0,0 & 0,0 & 1,1 & 0,0 & 1,0 & 1,0 & 1,1 & 1,1 & 0,1 \\
SP_2 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,0 & 1,0 & 1,1 & 0,1 & 0,0 \\
SP_3 & 0,0 & 1,0 & 0,0 & 1,0 & 0,1 & 0,0 & 0,0 & 1,0 & 0,1 & 0,0 \\
SP_4 & 1,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0 & 0,1 & 0,0 \\
SP_5 & 0,0 & 1,0 & 1,0 & 0,0 & 0,0 & 0,1 & 0,1 & 0,0 & 0,0 & 0,0 \\
SP_6 & 0,0 & 1,0 & 0,0 & 1,0 & 0,0 & 0,0 & 1,0 & 1,1 & 0,1 & 0,1 \\
SP_7 & 0,0 & 1,0 & 0,0 & 1,0 & 1,0 & 0,1 & 0,0 & 1,1 & 0,1 & 0,0 \\
SP_8 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,1 & 0,0 \\
SP_9 & 0,1 & 0,0 & 0,0 & 1,0 & 0,0 & 1,0 & 1,0 & 1,1 & 0,0 & 0,1 \\
SP_{10} & 0,0 & 1,0 & 1,0 & 1,0 & 0,1 & 1,0 & 1,0 & 1,1 & 0,1 & 0,0 \\
\hdashline
\text{Sum} & (1,1) & (5,0) & (2,0) & (8,1) & (1,2) & (3,2) & (6,1) & (8,6) & (1,8) & (0,3)
\end{array}
$$

Fig. 4. The Deadlocks Matrix for the sample problem.

Consequently, Rough Evacuation Priorities of robots can be decided by sorting the first elements of the sum in descending order, and Rough Occupation Priorities of robots can be decided by sorting the second elements of the sum in ascending order. If two or more robots have equal sum of elements, then they have equal motion priorities. For our example, the Evacuation Priorities (EP) and Occupation Priorities (OP) of the robots are worked out as below:

$$\boxed{\begin{array}{c} \{EP_4, EP_8\} > EP_7 > EP_2 > EP_6 > EP_3 > \{EP_9, EP_5, EP_1\} > EP_{10} \\ \{OP_2, OP_3\} > \{OP_1, OP_4, OP_7\} > \{OP_5, OP_6\} > OP_{10} > OP_8 > OP_9 \end{array}}$$

It should be noted that Rough Prioritization may fail to provide a comprehensive prioritization since it does not consider all decisive factors. Thus, further refinements are needed to finalize the motion priorities.

B. Exact Motion Prioritization

In order to obtain a finalized and exact motion priority scheme for the robots, both their start and goal locations and deviation classes must be taken into consideration. Table VII presents a number of simple rules for setting priorities. The first 3 rules are based on the fact that robots with lower Levels of starts (goals) are closer to the 'center' (i.e. Origin) of the tree, and so must evacuate as early as (occupy as late as) possible. The remaining rules are deduced directly from Table I.

TABLE VII
RULES FOR DETERMINING THE MOTION PRIORITIES OF ROBOTS

Rule	Condition			Priority
1	if	$l(s_i) > l(s_j)$	and $s_j \in Path(\mathrm{O}, s_i)$,	then $EP_j > EP_i$
2	if	$l(g_i) > l(g_j)$	and $g_j \in Path(\mathrm{O}, g_i)$,	then $OP_i > OP_j$
3	if	$l(s_i) > l(g_j)$	and $g_j \in Path(\mathrm{O}, s_i)$,	then $EP_i > OP_j$
4	if r_i = DR,	r_j = NDR,	and Deadlock Case = 1,	then $EP_i > EP_j$
5	if r_i = DR,	r_j = NDR,	and Deadlock Case = 1,	then $OP_j > OP_i$
6	if r_i = DR,	r_j = NDR,	and Deadlock Case = 2 to 7,	then $EP_i > EP_j$
7	if r_i = DR,	r_j = NDR,	and Deadlock Case = 2 to 7,	then $OP_j > OP_i$

Based on these rules the Evacuation and Occupying Priorities for the sample problem are further refined and finalized in Table VIII. As can be seen in the final priorities scheme, some robots may still have equal motion priorities (included in braces). This is because those robots are not involved in a deadlock. Also, note that some priorities obtained in Rough Prioritization may change after Exact Prioritization, as for EP_5 and EP_{10}.

TABLE VIII
EXACT MOTION PRIORITIES FOR THE SAMPLE PROBLEM

Type	Rule	Priorities
EP	1	$EP_8{>}EP_2{>}EP_5$; $EP_8{>}EP_2{>}EP_3{>}EP_{10}$; $EP_8{>}EP_1$; $EP_8{>}EP_4{>}EP_7{>}EP_6{>}EP_9$;
	4	$EP_8{>}EP_1$; $EP_8{>}EP_2$; $EP_8{>}EP_{10}$;
	6	$EP_4{>}EP_1$; $EP_9{>}EP_1$; $EP_7{>}EP_2$; $EP_9{>}EP_{10}$
OP	2	$OP_5{>}OP_9$; $OP_3{>}OP_9$; $OP_1{>}OP_{10}{>}OP_8{>}OP_9$; $OP_4{>}OP_9$; $OP_2{>}OP_8{>}OP_9$; $OP_7{>}OP_6{>}OP_9$;
	5	$OP_1{>}OP_8$; $OP_{10}{>}OP_8$; $OP_2{>}OP_8$;
	7	$OP_1{>}OP_4$; $OP_1{>}OP_9$; $OP_2{>}OP_7$; $OP_{10}{>}OP_9$;
EP – OP	3	$EP_9{>}OP_1$; $EP_6{>}OP_8$; $EP_7{>}OP_9$; $EP_3{>}OP_9$; $EP_1{>}OP_4$; $EP_5{>}OP_7$; $EP_{10}{>}OP_1$; $EP_1{>}OP_9$;

FINAL PRIORITY SCHEME

$$EP_8 > EP_4 > EP_7 > EP_2 > EP_6 > EP_3 > EP_9 > EP_1 > EP_{10} > EP_5$$
$$\{OP_2, OP_3\} > \{OP_1, OP_4, OP_7\} > \{OP_5, OP_6\} > OP_{10} > OP_8 > OP_9$$

V. DETERMINING TEMPORARY STATIONS

In this phase the vertices on which the deviating robots should temporarily reside are determined. For this purpose, two approaches can be adopted: (1) using the tree's existing vertices, or, (2) inserting new vertices into the tree.

For the first approach, the temporary stations must be selected such that the deviation distance is kept minimal, and the station should not intercept any NDR which will move later.

The second approach requires that the original C-space is wide enough near the insertion point to permit such an expansion. Also, to keep the graph's size as small as possible, these additional vertices should be inserted carefully and strategically so that the resulting tree is minimal in both size and required robotic moves. Vertices on which the new vertices must be inserted are called *Receiving Vertices* (RV). Depending on the class of the robots appeared in a deadlock situation, the new vertices are inserted based on either path priorities, or motion priorities of robots.

A. Vertex Insertion Based on Path Priorities

In resolving a deadlock situation where a DR and an NDR are in conflict, a new vertex should be annexed to the DR's path to let it deviate. Table IX presents some rules developed based on Table I and simple logic to identify Receiving Vertices.

TABLE IX
DETERMINING RECEIVING VERTICES BASED ON PATH PRIORITIES

Case	Condition	Receiving vertices
1	only r_i is DR	all in SP_i
2	only r_i is DR	all in SP_i
	only r_j is DR	$\{SP_i \cap SP_j\}$ except $\{s_i, g_i\}$
3	only r_i is DR	SP_i except s_i
	only r_j is DR	$\{SP_i \cap SP_j\}$ except $\{s_i, g_i\}$
4	only r_i is DR	SP_i except g_i
	only r_j is DR	$\{SP_i \cap SP_j\}$ except $\{s_i, g_i\}$
5	only r_i is DR	SP_i except $\{s_i, g_i\}$
	only r_j is DR	SP_j except $\{s_j, g_j\}$
6	only r_i is DR	$\{SP_i \cap SP_j\}$ except g_j
	only r_j is DR	$\{SP_i \cap SP_j\}$ except g_i
7	only r_i is DR	$\{SP_i \cap SP_j\}$ except s_j
	only r_j is DR	$\{SP_i \cap SP_j\}$ except s_i

B. Vertex Insertion Based on Motion Priorities

In a deadlock situation where two DRs are involved, a new vertex must be inserted for each robot to let it deviate. Based on the order of deviation, there can be different Receiving Vertices for each robot. The first new vertex is inserted somewhere on the path of the first deviating robot, and the location of the second new vertex is afterwards decided with respect to the first new vertex. Receiving Vertices are determined by using the Exact Evacuation Priorities of the robots and based on the type of deadlocks, as described in Table X.

The set of RVs for vertex insertion in the example problem are calculated in Table XI. Note that a new vertex is inserted on only one of the Receiving Vertices of each DR, depending on the availability of sufficient free space around (see Fig. 3).

TABLE X
DETERMINING RECEIVING VERTICES BASED ON MOTION PRIORITIES

Case	Condition	Receiving Vertices on SP_i	Receiving Vertices on SP_j
1	$EP_i > EP_j$	all	all
	$EP_j > EP_i$	all	all except $Path(s_i, g_j)$
2	$EP_i > EP_j$	all	all
	$EP_j > EP_i$	all	all except $Path(s_i, g_j)$
3	$EP_i > EP_j$	all	$\{SP_i \cap SP_j\}$ except g_i
	$EP_j > EP_i$	all except s_i	all except g_j
4	$EP_i > EP_j$	all except g_i	all
	$EP_j > EP_i$	all	$\{SP_i \cap SP_j\}$ except s_i
5	$EP_i > EP_j$	all except g_i	all except s_j
	$EP_j > EP_i$	all except s_i	all except g_j
6	$EP_i > EP_j$	all	$\{SP_i \cap SP_j\}$ except g_i
	$EP_j > EP_i$	$\{SP_i \cap SP_j\}$ except g_i	all
7	$EP_i > EP_j$	$\{SP_i \cap SP_j\}$ except s_j	all
	$EP_j > EP_i$	all	$\{SP_i \cap SP_j\}$ except s_i

TABLE XI
IDENTIFYING THE RECEIVING VERTICES FOR THE SAMPLE PROBLEM

Robots	Deadlock Case	Deviating Robot	Receiving Vertices
$\{r_1, r_4\}$	4	r_4	RV(r_4) = {A, C}
$\{r_1, r_8\}$	1	r_8	RV(r_8) = {A, C, E}
$\{r_1, r_9\}$	3	r_9	RV(r_9) = {G, E, C, A}
$\{r_2, r_7\}$	7	r_7	RV(r_7) = {A, C, E}
$\{r_2, r_8\}$	1	r_8	RV(r_8) = {A, C, E}
$\{r_8, r_{10}\}$	1	r_8	RV(r_8) = {A, C, E}
$\{r_9, r_{10}\}$	6	r_9	RV(r_9) = {A, C, E}

$\{r_4, r_8\}$	7	r_4 and r_8
		$EP_8 > EP_4$
			RV(r_8) = {A}
			RV(r_4) = {A, C, B}

$\{r_6, r_8\}$	2	r_6 and r_8
		$EP_8 > EP_6$
			RV(r_8) = {A, C, E}
			RV(r_6) = {G, E, C, A, I, L}

$\{r_7, r_8\}$	4	r_7 and r_8
		$EP_8 > EP_7$
			RV(r_8) = {A, C}
			RV(r_7) = {E, C, A, I, L, M}

$\{r_8, r_9\}$	3	r_9 and r_8
		$EP_8 > EP_9$
			RV(r_8) = {A, C, E}
			RV(r_9) = {A, C}

FINAL DECISION

RV(r_4) = {A, C} ∩ {A, C, B} = {A, C}, RV(r_6) = {G, E, C, A, I, L},
RV(r_7) = {A, C, E} ∩ { E, C, A, I, L, M} = {A, C, E},
RV(r_8) = {A, C, E} ∩ {A, C} ∩ {A} = {A},
RV(r_9) = {G, E, C, A} ∩ {A, C, E} ∩ {A, C} = {A, C}.

C. Generation of the Final Plan

After deciding the motion priorities and Receiving Vertices, the final Plan (Table XII) can now be generated according to this procedure: (1) Move the robots in order of their Evacuation Priorities (DRs to their RVs and NDRs to goals), and (2) Move DRs to their goals in order of their Occupation Priorities. The total number of moves in the Plan can be directly calculated by:

$$C = \sum_{i=1}^{m} \left| SP_i \right| + 2\left| \mathrm{DR} \right| - m = 50 + 2 \times 5 - 10 = 50 \cdot$$

TABLE XII
THE FINAL PLAN OF ROBOTS MOTIONS FOR THE SAMPLE PROBLEM

$\boldsymbol{r_8}$:{A→A$_1$};	$\boldsymbol{r_4}$:{C→A→A$_2$};	$\boldsymbol{r_7}$:{E→C→C$_1$};	$\boldsymbol{r_2}$:{ I→A→C→E→F };
$\boldsymbol{r_6}$:{G→E→C→A→I→I$_1$};	$\boldsymbol{r_3}$:{J→I→A→C→D};		$\boldsymbol{r_9}$:{H→G→E→C→C$_2$};
$\boldsymbol{r_1}$:{B→A→C→E→G→ H};			$\boldsymbol{r_{10}}$:{K→J→I→A→C→E→G};
$\boldsymbol{r_5}$:{M→L→I→J};	$\boldsymbol{r_7}$:{C$_1$→C→A→I→L→M};		$\boldsymbol{r_4}$:{A$_2$→A→B};
$\boldsymbol{r_6}$:{I$_1$→I→L};	$\boldsymbol{r_8}$:{A$_1$→A→C→E};		$\boldsymbol{r_9}$:{C$_2$→C→A};

VI. DISCUSSION AND CONCLUSION

In this paper a new decoupled algorithm is developed for solving multi robot motion planning problems. After that all *m* robots are checked pairwise for probable deadlocks in $O(m^2)$ time, the deviating class of the robots are identified, and their Path and Motion Priorities are determined through rough and exact levels, all in linear time. The tree can then be expanded to minimize the moves of deviating robots. The final Plan is then generated based on Evacuation and Occupation priorities.

To evaluate the efficacy of the presented method, 26 problems on trees with different number of vertices and robots were designed and solved, starting from 3 robots on 6 vertices to 15 robots on 18 vertices. Comparisons in Fig. 5 show that the presented prioritization method together with inserting new vertices (the lower curve) produced solutions with fewer moves than optimal solutions on original (not appended) trees.

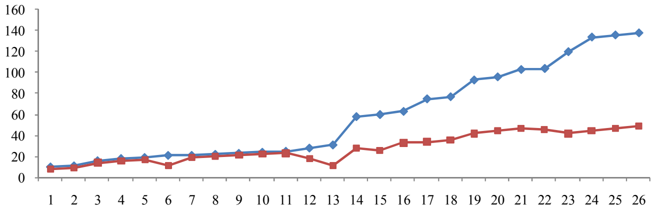


Fig. 5. Number of moves for the 26 test problems by both methods.

REFERENCES

[1] G. Calinescu, A. Dumitrescu and J. Pach, "Reconfigurations in graphs and grids," *Springer Lecture Notes in Comp. Sci.*, Vol. 3887, 2006, pp. 262-273.

[2] C. Belta and R.V. Kumar, "Motion generation for groups of robots: a centralized, geometric approach," in *Proc. ASME DETC*, Montreal, 2002.

[3] M. Saha, P. Isto, "Multi-robot motion planning by incremental coordination," in *Proc. IEEE IROS*, Beijing, pp. 5960-5963, 2006.

[4] J. van den Berg and M. Overmars, "Prioritized motion planning for multiple robots," in *Proc. IEEE IROS*, Edmonton, pp. 2217-2222, 2005.

[5] S.J. Buckley, "Fast motion planning for multiple moving robots," in *Proc. IEEE Int. Conf. Rob. and Autom.*, Scottsdale, AZ, pp. 322-326, 1989.

[6] C. Ferrari, E. Pagello, J. Ota, and T. Arai, "Multi robot motion coordination in space and time," *Rob. and Auton. Syst.*, Vol. 25, 1998, pp. 219-229.

[7] M. Bennewitz, W. Burgard and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Rob. and Auton. Syst.*, Vol. 41, No. 2, 2002, pp. 89-99.

[8] R. Regele and P. Levi, "Cooperative multi-robot path planning by heuristic priority adjustment," in *Proc. IEEE IROS*, Beijing, pp. 5954-5959, 2006.

[9] G.A.S. Pereira, A.K. Das, R.V. Kumar, M.F.M. Campos, "Decentralized motion planning for multiple robots subject to sensing and communication constraints," in *Proc. 2nd Int. Workshop on Multi-Rob. Sys.* 2003, pp. 267-278.

[10] M.R.K. Ryan, "Exploiting subgraph structure in multi-robot path planning", *J. Artificial Intell. Research*, Vol. 31, pp. 497-542, 2008.

[11] S. Onn, and E. Sperber, "Social network coordination and graph routing," *Networks*, Vol. 41, No. 1, 2003, pp. 44-50.

[12] M. Peasgood, J. McPhee, and C.M. Clark. "Complete and scalable multi-robot planning in tunnel environments," in *Proc. 1st IFAC Workshop on Multi-Vehicle Systems*, Oct. 2006.

[13] E. Masehian and A.H. Nejad, "Solvability of multi robot motion planning problems on trees," in *Proc. IEEE IROS,* St. Louis, USA, 2009.